

## Übungsblatt 6, Aufgabe 2 c

```
let rec iter (n:int) (f:int → int) (x:int) : int =
  if n = 0 then x else f (iter (n - 1) f x)
in iter 2 (λ x:int. x * x) 3
```

steht für

```
let iter = rec iter:int → (int → int) → int → int.
  λ n:int. λ f:int → int. λ x:int. if = n 0 then x else f (iter (- n 1) f x)
in iter 2 (λ x:int. * x x) 3
```

→ let iter = λ n:int. λ f:int → int. λ x:int. if = n 0 then x else f ((rec iter ... ) (- n 1) f x)  
in iter 2 (λ x:int. \* x x) 3

mit Regel (UNFOLD)

→ (λ n:int. λ f:int → int. λ x:int. if = n 0 then x else f ((rec iter ... ) (- n 1) f x)) 2 (λ x:int. \* x x) 3

mit Regel (LET-EXEC)

→ (λ f:int → int. λ x:int. if = 2 0 then x else f ((rec iter ... ) (- 2 1) f x)) (λ x:int. \* x x) 3

mit Regel (BETA-V)

→ (λ x:int. if = 2 0 then x else (λ x:int. \* x x) ((rec iter ... ) (- 2 1) (λ x:int. \* x x) x)) 3

mit Regel (BETA-V)

→ if = 2 0 then 3 else (λ x:int. \* x x) ((rec iter ... ) (- 2 1) (λ x:int. \* x x) 3)

mit Regel (BETA-V)

→ if false then 3 else (λ x:int. \* x x) ((rec iter ... ) (- 2 1) (λ x:int. \* x x) 3)

mit Regel (BOP)

→ (λ x:int. \* x x) ((rec iter ... ) (- 2 1) (λ x:int. \* x x) 3)

mit Regel (COND-FALSE)

→ (λ x:int. \* x x) ((λ n:int. λ f:int → int. λ x:int. if = n 0 then x else f ((rec iter ... ) (- n 1) f x)) (- 2 1) (λ x:int. \* x x) 3)

mit Regel (UNFOLD)

→ (λ x:int. \* x x) ((λ n:int. λ f:int → int. λ x:int. if = n 0 then x else f ((rec iter ... ) (- n 1) f x)) 1 (λ x:int. \* x x) 3)

mit Regel (BOP)

→ (λ x:int. \* x x) ((λ f:int → int. λ x:int. if = 1 0 then x else f ((rec iter ... ) (- 1 1) f x)) (λ x:int. \* x x) 3)

mit Regel (BETA-V)

→ (λ x:int. \* x x) ((λ x:int. if = 1 0 then x else (λ x:int. \* x x) ((rec iter ... ) (- 1 1) (λ x:int. \* x x) x)) 3)

mit Regel (BETA-V)

→ (λ x:int. \* x x) (if = 1 0 then 3 else (λ x:int. \* x x) ((rec iter ... ) (- 1 1) (λ x:int. \* x x) 3))

mit Regel (BETA-V)

→ (λ x:int. \* x x) (if false then 3 else (λ x:int. \* x x) ((rec iter ... ) (- 1 1) (λ x:int. \* x x) 3))

mit Regel (BOP)

→ (λ x:int. \* x x) ((λ x:int. \* x x) ((rec iter ... ) (- 1 1) (λ x:int. \* x x) 3))

mit Regel (COND-FALSE)

→ (λ x:int. \* x x) ((λ x:int. \* x x) ((λ n:int. λ f:int → int. λ x:int. if = n 0 then x else f ((rec iter ... ) (- n 1) f x)) (- 1 1))

mit Regel (UNFOLD)

→ (λ x:int. \* x x) ((λ x:int. \* x x) ((λ n:int. λ f:int → int. λ x:int. if = n 0 then x else f ((rec iter ... ) (- n 1) f x)) 0 (λ x:int. \* x x) 3))

mit Regel (BOP)

→ (λ x:int. \* x x) ((λ x:int. \* x x) ((λ f:int → int. λ x:int. if = 0 0 then x else f ((rec iter ... ) (- 0 1) f x)) (λ x:int. \* x x) 3))

mit Regel (BETA-V)

→ (λ x:int. \* x x) ((λ x:int. \* x x) ((λ x:int. if = 0 0 then x else (λ x:int. \* x x) ((rec iter ... ) (- 0 1) (λ x:int. \* x x) x)) 3))

mit Regel (BETA-V)

→ (λ x:int. \* x x) ((λ x:int. \* x x) (if = 0 0 then 3 else (λ x:int. \* x x) ((rec iter ... ) (- 0 1) (λ x:int. \* x x) 3))))

mit Regel (BETA-V)

→  $(\lambda x:\mathbf{int}. * x x) ((\lambda x:\mathbf{int}. * x x) (\mathbf{if\ true\ then\ 3\ else\ } (\lambda x:\mathbf{int}. * x x) ((\mathbf{rec\ iter\ \dots}) (- 0 1) (\lambda x:\mathbf{int}. * x x) 3)))$   
mit Regel (BOP)

→  $(\lambda x:\mathbf{int}. * x x) ((\lambda x:\mathbf{int}. * x x) 3)$   
mit Regel (COND-TRUE)

→  $(\lambda x:\mathbf{int}. * x x) (* 3 3)$   
mit Regel (BETA-V)

→  $(\lambda x:\mathbf{int}. * x x) 9$   
mit Regel (BOP)

→  $* 9 9$   
mit Regel (BETA-V)

→  $81$   
mit Regel (BOP)